Implementing LORA framework on a Merged Diffusion Model for Image Synthesis

Abstract

In this paper, we have tried to explore and understand the working of state-of-art Diffusion models which are being widely used for image synthesis. In addition to this, we have also explored the area for merging two diffusion models with different parameters setting and weightage to build a merged diffusion model which exhibits the features of both the models. To enhance the experimentation further we have tried to implement the famous LORA framework on the merged diffusion model which helps in fine tuning of the model, along with reducing the computational power required to finetune a stable diffusion model in the general sense.

1. Introduction

One of the computer vision areas with the most spectacular recent advancements, but also one with the highest computational demands, is image synthesis. Diffusion models have swept the globe since the release of Dall-E 2 and Google's Imagen, Stable Diffusion, and Midjourney, inspiring innovation and pushing the limits of machine learning. The state-of-the-art in class-conditional image synthesis] and super-resolution has recently been defined by diffusion models, which are constructed from a hierarchy of denoising autoencoders and achieve impressive results in image synthesis and beyond. In contrast to other kinds of generative models, even unconditional DMs can be applied easily to tasks like inpainting and colorization. Given the variety of methods and approaches available, fine-tuning stable diffusion models can be a difficult and time-consuming process.

In this paper we try to leverage the LoRA framework for Pre-training and Transfer Learning of stable diffusion models. Pre-trained language models can capture general knowledge and patterns from vast amounts of text data. By leveraging transfer learning, we can fine-tune these pre-trained models on specific diffusion-related tasks like image generation, allowing them to understand domain-specific language and dynamics.

2. Related Work

There have been multiple research and experimentation done on checkpoint model merging of stable diffusion models and finetuning models using LoRA and other methods. For our approach we took the reference of the well documented hugging face diffusion and stable diffusion libraries and have used scripts for training that have been provided in their github.

3. Data

For the Data, we have used a self-created dataset of 112 images with their respective text captionings, the dataset is publicly available on huggingface's dataset API and can be used for other projects as well just by using the link for the dataset. The subject of our dataset is a character

from a famous Japanese anime named Soryu Asuka Langley, we have gone on more detail about this in the later parts of the report.

4. Methods

Diffusion Models

With record-breaking performance in numerous applications, such as image synthesis, video generation, and molecule design, diffusion models have become a potent new family of deep generative models. It is a kind of Generative model which creates new data based on training data. Other generative models include GANs (Generative adversarial networks), VAEs (variational autoencoders) and flow-based models.

Diffusion models learn to recover the data by reversing this noise-adding process after first destroying the training data by adding noise. To put it another way, diffusion models can create coherent images out of noise. Diffusion models learn by introducing noise to images, which the model later masters the removal of. To produce realistic images, the model then applies this denoising technique to random seeds.

What makes them so unique?

Their capacity to produce extremely realistic imagery and better replicate the distribution of actual images than GANs is the most apparent elucidation.

The fundamental idea- The fundamental idea behind diffusion modelling is that if we can create a learning model that can recognize the systematic loss of information caused by noise, we should be able to reverse the process and pull the information back from the noise. In that it attempts to optimize an objective function by first projecting the data onto the latent space and then recovering it to its initial state, this concept is comparable to VAEs. However, the system aims to model a series of noise distributions in a Markov Chain rather than learning the data distribution, and it "decodes" the data by undoing/denoising it in a hierarchical manner.

LoRA Framework

LoRA stands for **Low Rank Adaptation of Large Language Models**. In a study titled LoRA, the low-rank property of attention mechanisms in large language models is used to propose an effective method for modifying these models to fit new domains or tasks. Many language models, including transformers, depend heavily on the attention mechanism, which enables them to concentrate on pertinent segments of the input sequence when producing the output. It is mainly used to fine tune stable diffusion model. However, we already have different methods like *Dreambooth* and *textual inversion*; so why do we need LoRA? The answer lies in the amazing trade-off provided by LoRA between file size and training power. It has a much more manageable file size of 2 to 200 MB, and the training power is satisfactory.

Figure 1: Yae Miko | Realistic Genshin LORA

How does LoRA work?

The cross-attention layers, which are a crucial component of Stable Diffusion models, are subject to minor changes in LoRA. It is the area of the model where the prompt and the image converge.

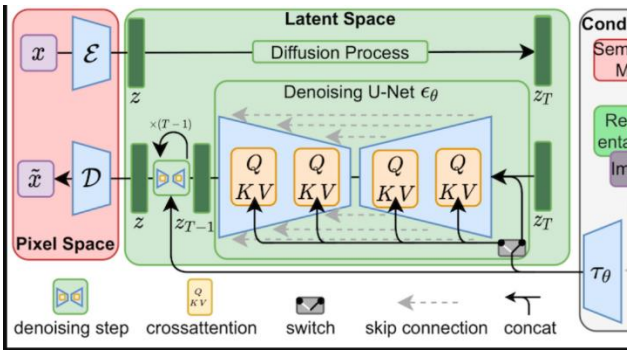

Figure 2. LORA fine-tunes the cross-attention layers (the QKV parts of the U-Net noise predictor). (Figure from Stable Diffusion paper.)

A cross-attention layer's weights are organized in matrices. Just like an Excel spreadsheet, matrices are nothing more than a collection of numbers arranged in columns and rows. Through the addition of its weights to these matrices, a LoRA model adjusts a model. LoRA achieves smaller file size by breaking a matrix down into two smaller matrices (low rank).

Here is the example taken from [Stable Diffusion Art]

Let's say the model has a matrix with 1,000 rows and 2,000 columns. That's 2,000,000 numbers (1,000 x 2,000) to store in the model file. LoRA breaks down the matrix into a 1,000-by-2 matrix and a 2-by-2,000 matrix. That's only 6,000 numbers (1,000 x 2 + 2 x 2,000), 333 times less. That's why LoRA files are a lot smaller.

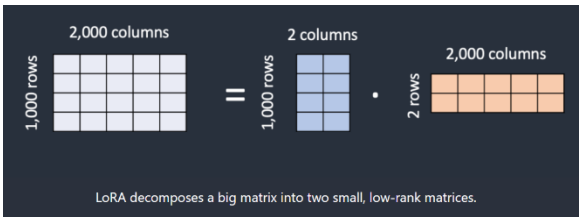

Figure 3. matrix decomposition

By fine-tuning the language model with fewer additional parameters, LoRA can adapt the language model more effectively to new tasks or domains. This can reduce the need for computational resources and facilitate the deployment of large language models across a range of applications. Overall, LoRA is a method that makes it easier and more computationally effective to adapt large language models for tasks or domains.

Leveraging LoRA for Pre-training and Transfer Learning:

1. **Pre-trained Image Generation Model:** First, we start by pre-training a powerful image generation model on a large dataset of images. The model is a synthesis model. This pre-training step allows the model to learn the

underlying patterns and features present in images.

2. **Image Diffusion Process:** Next, we define our diffusion model for image synthesis. The diffusion process aims to generate images by iteratively applying noise and modifying the pixel values. This process allows us to generate images step by step, capturing the distribution of the data effectively.
3. **LoRA for Adaptation:** Now, here's where we integrate LoRA. We'll exploit the low-rank property of the model's internal attention mechanisms to make the adaptation process more efficient. LoRA allows us to fine-tune the model for the image generation with fewer parameters, which can save computational resources and training time.
4. **Fine-tuning and Transfer Learning:** Using the LoRA-adapted model as a starting point and fine-tuning it on a dataset. This fine-tuning process helps the model learn the specific features and characteristics of the domain while retaining the general knowledge from pre-training. Transfer learning is a crucial aspect of this step, as it allows us to leverage the knowledge learned during pre-training for the new image synthesis task.
5. **Diffusion Based Image Generation:** Finally, using the adapted and fine-tuned image generation model in conjunction with the diffusion process to generate images. During the diffusion process, the model modifies the pixel values iteratively, generating high-quality images based on the diffusion dynamics learned from the dataset.

5. Experimentation

Merging of Models

One of the techniques we explored in the vast field of Stable Diffusion was to merge two checkpoint models. The way the merger works is to iterate over the "State_dict" of two models one by one and combine them in a set ratio (say alpha). The State_dict is nothing but a simple dictionary object in pytorch where the weights of the models are saved. For our example, we merged the weights by using the following equation:

$$\theta_0 = (1 - \alpha)\theta_0 + \alpha.\theta_1$$

Where θ_0 is the weight of the first model and θ_1 is the weight of the second model. Conveniently, we do leave out merging the weights of the VAE model as it has shown to make inferences worse for anime models such as ours. The models that we merged with were AbyssOrangeMix by WarriorMama777, and waifu-diffusion by hakurei. Both models excel as anime models, though they do have their own unique art style which is evident when we give them the same prompt. Our goal was to get a nice mix of the two models' art styles and for that we took an alpha of 0.7, giving 70% weightage to AbyssOrangeMix

and 30% to waifu-diffusion. The higher weightage is done as the former model does give higher quality results than the latter.

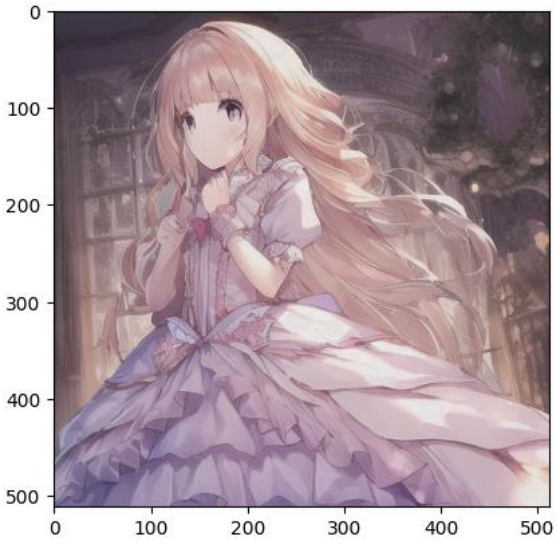

Figure 4. Inference for AbyssOrangeMix

The prompt that was given to all three models was “An image of a girl wearing princess clothes”, and as evident in Fig. 4, the model was able to infer a nice anime style image of a girl wearing a dress.

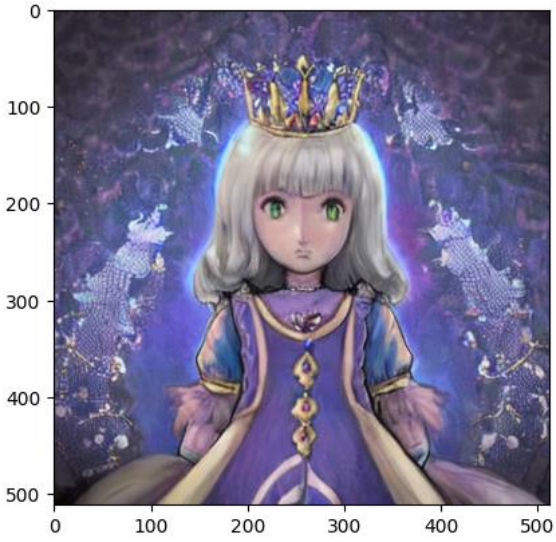

Figure 5. Inference for waifu-diffusion

While the prompt given to waifu-diffusion was the same, it evidently gives us a relatively bad image, maybe this is due to the prompt not having any “high quality”,” good quality”, etc. as the prompts but nonetheless we do see a bad inference.

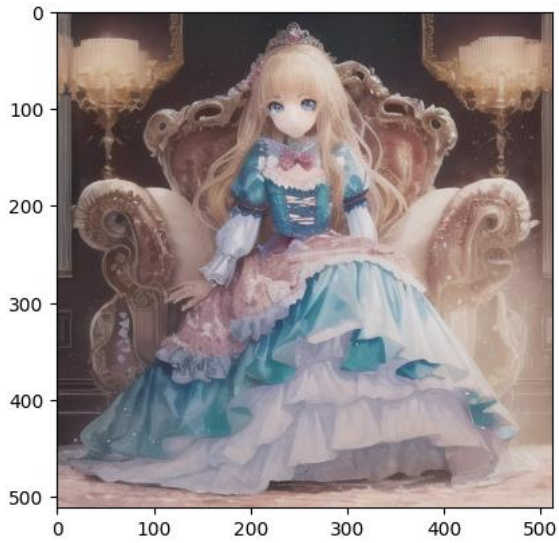

Figure 6. Inference for the merged model

Lastly, we merged the two above models and ran an inference test with the same prompt as the above three and we do get an interesting result. While we do have the art style of AbyssOrangeMix with its other features due to the high weightage given to the model, we also see a bit more colorful image probably due to the features of waifu-diffusion coming into play.

LoRA finetuning of the Merged Model

The next obvious step to our experiment was to finetune the merged model, for this we used the LoRA finetuning approach

that has been explained earlier in the paper. The subject we chose our finetuning to be done on is a famous character from a 90s Japanese anime named “Soryu Asuka Langley” or just Asuka for short. Our dateset consists of 112 images of Asuka with appropriate captioning for each of the images.

Figure 7. A picture of Asuka for reference

There are two ways to go about using LoRA model training to finetune a stable diffusion model when it comes to using hugging face’s libraries, the first way is to use the text_to_image approach which takes a dataset of images with corresponding text captions (mostly BLIP captioning) to train it both on the texts and images for better inference results. Before the training was done however, the merged model first needed to be converted into the pipeline format that hugging face requires, this was done easily with an accelerate script. The training was done with a total of 1120 optimization steps which accounted for 40 epochs. After the training is completed, we get a very small model file of just a few MBs.

Next, we use the second approach of training a LoRA model which is the Dreambooth LoRA method. While this method is originally to finetune Dreambooth models, nothing stops us from using this method to train it on any model. One drawback is that there is no text caption training for the model or at least none that we could find, this does impact our inference and maybe in the future we explore other methods of LoRA training such as Automatic1111 and Kohya’s methods to train LoRAs.

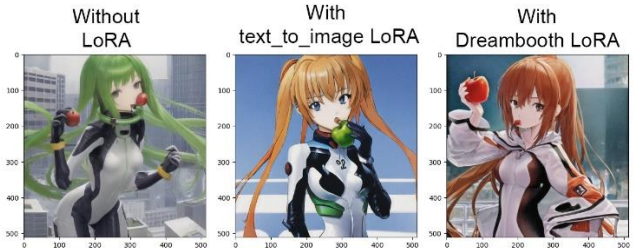

Figure 8. Inference test for all three approaches.

Lastly, the only thing that was logically left for us to do was to run an inference test on the model without LoRA, with text_to_image LoRA and with Dreambooth LoRA. The prompt that was given to all three were “1girl, image of girl eating apple, plugsuit 02”. As we can see in the figure, the model without LoRA gives us an image of a random green haired girl eating an apple, while the other two models with LoRA give us a blonde hair girl eating an apple. It is also interesting to note that the text_to_image LoRA gives us an inference that looks the most like Asuka out of all the three, and while there is no metric to compare the three, it can be said that text_to_image LoRA works the best for our approach.

6. Conclusion

Stable Diffusion's full model fine-tuning used to be time-consuming and challenging, which is in part why faster, simpler techniques like Dreambooth or Textual Inversion have gained popularity. On a unique dataset, LoRA makes it much simpler to fine-tune a model.

This method offers following advantages:

- As was already mentioned, training goes much more quickly.
- Less computing power is required.
- The size of trained weights is incredibly small. We can save the weights for the new layers as a single file that is about 3 MB in size because the original model is frozen, and we inject new layers to be trained. This is considerably smaller than the UNet model's initial dimensions!

Hence, we can conclude that the out of the three models, text_to_image LoRA performed comparatively better than the rest. Since as of now we don’t have any measuring parameters, we infer this on the basis of how closely the model is able to represent or exhibit the features of a real image.

References:

1. Hu, E. J. (n.d.). *LoRA: Low-Rank Adaptation of Large Language Models*. https://doi.org/arXiv:2106.09685v2
2. Rombach, R. (n.d.). *High-Resolution Image Synthesis with Latent Diffusion Models*. https://doi.org/10.48550/arXiv.2112.10752
3. Yang, L. (n.d.). *Diffusion Models: A Comprehensive Survey of Methods and Applications*. https://doi.org/arXiv:2209.00796v10
4. Using LoRA for Efficient Stable Diffusion Fine-Tuning (huggingface.co)
5. What are LoRA models and how to use them in AUTOMATIC1111 - Stable Diffusion Art (stable-diffusion-art.com)
6. Diffusion Models: A Practical Guide | Scale AI
7. Diffusion Models Made Easy. Understanding the Basics of Denoising… | by J. Rafid Siddiqui, PhD | Towards Data Science
8. Introduction to Diffusion Models for Machine Learning (assemblyai.com)
9. Maximizing Your Stable Diffusion Fine-Tuning: A Framework (substack.com)